# OpenCV WebCam Applications in an Arduino-based Rover *

Valeria Loscrí[1], Nathalie Mitton[1], Emilio Compagnone[2]

[1] Inria, France [2] University of Calabria, Italy

**Abstract.** In this work we design and implement Arduino-based Rovers with characteristics of re-programmability, modularity in terms of type and number of components, communication capability, equipped with motion support and capability to exploit information both from the surrounding and from other wireless devices. These latter can be homogeneous devices (i.e. others similar rovers) and heterogeneous devices (i.e. laptops, smartphones, etc.). We propose a Behavioral Algorithm that is implemented on our devices in order to supply a proof-of-concept of the effectiveness of a Detection task. Specifically, we implement the "Object Detection" and "Face Recognition" techniques based on OpenCV and we detail the modifications necessary to work on distributed devices. We show the effectiveness of the controlled mobility concept in order to accomplish a task, both in a centralized way (i.e. driven by a central computer that assign the task) and in a totally distributed fashion, in cooperation with other Rovers. We also highlight the limitations of similar devices required to accomplish specific tasks and their potentiality.

**Keywords:** Rover, OpenCV, WebCam applications

## 1 Introduction

In the last few years, we have witnessed a technological development in robots, computing and communications, that has allowed the design of Network Robotic Systems (NRS) [14]. A NRS is composed by a robotic unit, able to communicate and cooperate with both other similar units and other interconnected devices. A very interesting feature of this type of systems is the possibility to "use" the same robots to accomplish different and complex tasks and the same set of robots may perform the same task in different conditions [10]. There have been increasing interests in deploying a team of robots able to cooperate and to self-coordinate, to fulfill also complex tasks such as target tracking [3], disaster recovery [8], etc. NRS concept will marry in a very natural fashion with the concept of Swarm Robotics, where many robotic units are considered to cooperatively accomplish very complex tasks [11] [13]. Beni [2] provides a very effective and precise definition of a swarm of robots : "The group of robots is not just a group. It has

* This work has been partially supported by the FP7 VITAL.

some special characteristics, which are found in swarms of insects, that is, decentralized control, lack of synchronization, simple and (quasi) identical members". In this work, we overcome the concept of members that have to be similar by considering devices that are able to communicate with any interconnected node. In any case, we outline how the devices are able to cooperate and self-coordinate also without a central unit that drives them.

In this paper, we build an Arduino-based platform, that presents some important characteristics such as re-programmability, modularity in terms of type and number of components and communication capability. Our devices are also equipped with motion capability and are able to exploit the concept of controlled mobility, by reaching specific target locations. Our robots can leverage information both from the surrounding and from other interconnected devices (i.e. others similar robots, smartphones, laptops, etc.). Based on the acquired information, our devices will behave in a specific fashion, in order to accomplish a specific task assigned either by a central unit (e.g. a central computer where a human may ask the accomplishment of specific tasks) or by another robot. These actions are realized through a very simple Behavioral Algorithm that will be implemented onto the devices and will be tested in different scenarios, in order to verify the effectiveness and to highlight the critical aspect of the system. The robot will react based on external stimulus, acquired through a WebCam or communication equipment. In order to make the analysis of the acquired images feasible, we referred to a well-known vision tool, *Open Source Computer Vision Library*, OpenCV.

The main contributions of this work can be summarized as follows:

- Modifications of specific tools such as OpenCV libraries originally developed for more powerful devices (e.g. computer) to adapt them to distributed and constrained environments;
- Development of a testbed on real Arduino-based platforms;
- Proposal of a very simple Behavioral Algorithm to be implemented as a proof-of-concept of the developed tools and platform.

The rest of the paper is organized as follows. Section 2 describes the simple Behavioral Algorithm and the basic rules the robot accomplishes for searching target. Section 3 presents the tools we use to acquire data and how to exploit this information. Section 4 details how we modified existing tools in order to allow effective actions in a distributed and constrained environment. Section 5 presents the platform setup and implementation of the Behavioral Algorithm integrated with the modified libraries. Finally, we conclude this work in Section 6.

## 2 The Behavioral Algorithm

This section details our Behavior Algorithm for mobile robots. The main goal is to search for possible targets, identify them and thus, cover them by having robots reach them, in a distributed way. Each robot runs the same algorithm

independently of the others and cooperate to fulfill a list of tasks, *a list of targets to reach*.

First of all, it is worth defining what a target is. In this context, a target represents an object with specific characteristics in terms of shape and/or color, that has been detected by the robot. The specific characteristics are defined by a human controller through a central computer, but our project can be easily modified, by including a dynamic definition of the target also defined by other inter-connected devices. Initially, each robot is assigned with a target. A same target can be assigned to several robots.

The algorithm is mainly composed of two phases: the Searching Phase and the Approaching Phase. During both phases, an underlying obstacle avoidance process is running. It is run on every robot when moving. More details about the obstacle avoidance implementation are given in Section 4.3. The details of the Behavioral Algorithm are given in the following pseudo-code, on Algorithm 1. Our algorithm will terminate when all tasks assigned to the robot are completed.

The robot starts with the task on top of its list. It first enters the Searching Phase (Lines 19-37 in Algo. 1) which consists in locating the target. If the robot does not identify the pre-assigned target in its environment (in its vision field), it moves forward for an arbitrary time $\Delta t$ (e.g 1 or 2 sec.) by following a Random Way Path (RWP) [6]. More precisely, it travels a prefixed distance then it stops and checks for the presence of the target within its 'new' environment. It then repeats this process while the target is not identified. Once the target is identified, the robot switches to the Approaching Phase (Lines 4-19 in Algo. 1). The robot simply heads to the direction of the target till either reaching it or realizing that the target is a "false positive", due to for example to lights. In order to detect these "false positives", a specific mechanism is set up to allow the robot to make a new search without being influenced by the previous results as described in Section 4.2. It then switches back to the Searching Phase. Once the target is reached, the robot advertises other robots though its communication channel. Upon reception of this message, other robots abort their task to move to the next one. This latter phase can be revisited as a dynamic change of the task, based on the input that one of them broadcasts.

## 3 Background: software and hardware used in this work

In this section, we give a brief description of the hardware and software tools later used in this paper for our different implementations.

### 3.1 Hardware

In order to evaluate our contributions, we use the following hardware platform. An Arduino module [1] is set on an aluminum robotic platform with 4 wheels. Arduino allows us to re-program our node and to make it able to interact with

**Algorithm 1** Behavioral Algorithm

• **Local variables:** TargetFound = FALSE; TaskCompleted = FALSE;

```
1: while the task list is not empty do
2:     Move to next task on the list
3:     while (TargetFound = FALSE) AND (TaskCompleted = FALSE) do
4:         {Searching Phase}
5:         while (TargetFound = FALSE) AND (TaskCompleted = FALSE) do
6:             Δt ← Random().
7:             Move forward for Δt at speed s.
8:             Listen to other robots.
9:             if Target reached by another robot then
10:                TaskCompleted← TRUE.
11:            else
12:                Scan environment.
13:                if Target identified then
14:                    TargetFound←TRUE.
15:                else
16:                    Change direction.
17:                end if
18:            end if
19:        end while
           {Approaching Phase}
20:        while (TargetFound = TRUE) AND (TaskCompleted = FALSE) do
21:            Δt ← Random()
22:            Move toward target for Δt at speed s
23:            if Target is reached then
24:                TaskCompleted← TRUE.
25:                Advertise other robots.
26:            else
27:                Listen to other robots.
28:                if Target reached by another robot then
29:                    TaskCompleted← TRUE.
30:                else
31:                    Check false positive.
32:                    if False positive identified then
33:                        TargetFound← FALSE.
34:                    end if
35:                end if
36:            end if
37:        end while
38:    end while
39: end while
```

the external environment. Our Arduino module is a UNO rev3 which main characteristics are a microcontroller ATmega328, a 32B flash memory with a 16MHz clock [1] that we extended with an IMU board, a motor board and a PING sensor.

Since our devices are equipped with motion capabilities, we apply on them an ultrasound sensor in order to avoid obstacles (See Section 4.3). Moreover, we consider a WebCam to exploit images for task accomplishment. We used Open Source Computer Vision Library, OpenCV [12] libraries together with Computer Vision in order to "extract" meaningful data from the images. A very important component of our devices is the Inertial Movement Unit (IMU), a platform with an accelerometer, magnetometer and gyroscope. The IMU makes the Rover navigable. In effect, there exist other "easier" and well-performing

[1] http://store.arduino.cc/index.php?main_page=product_info&products_id=195&language=en

solutions to realize the "navigation" function, such as stepper [7], but the ratio cost/efficiency of our solution is better than the other available.

Since we chose a cheap micro controller, each rover has only limited computing capacity and is not able to run the tasks we address. Therefore, we embed a miniPC MK802II in our robots, that we modified in order to use a Linux platform instead of the Android available. The choice of this OS (Operating System) is related to the possibility to better control the platform with high level program languages, to increase the computational capability and to control external devices (i.e. those already mentioned), through an USB interface.

Wifi miniPC cardboard is used for peer-to-peer communications.

### 3.2 Software

In order to make our devices able to perform some specific tasks, we focused on Object Detection techniques. More specifically, we consider a Computer Vision library, named OpenCV [12]. OpenCV has been developed by Intel and supplies a set of high level functions to acquire images and computer vision in real time. An important feature of OpenCV is that it allows the execution on different platforms (Linux, windows, etc). We mainly focus on two specific techniques:

- *Face Recognition* based on the Cascade Classifiers method;
- *Object Detection* based on the HSV Model [5].

In the following, we detail these two techniques.

**Object Detection via HSV** Hue, Saturation and Value (HSV) consists in a re-deployment of points of the RGB Cartesian Space into a cylindric space. The main goal of this operation is a more intuitive analysis of the colors in its components, since in the RGB, the analysis is very complex. In Figure 1 we show how the HSV model works. In the figure, in each cylinder, the angle around the central axis represents the *tonality*, the distance from the central axis is the *saturation*, and the distance from the basis of the cylinder is the *value*.

Figure 2 shows the difference from an optical point of view obtained when we apply the HSV model on an RGB model. The evaluation of this difference has been very important for the usage of the HSV model as Object Detection. In order to make possible the detection operation, it is necessary to consider a color filtering phase based on the function inRange of OpenCV.

This filtering phase is paramount in the Object Detection, since we consider the form-color combination. It is worth noticing that the exact values for color filtering are not known and for that it is necessary the use of a specific tool that allow the extraction of HSV values from an image (see Figure 3).

It is also important to notice that $H$ channels range from 0 to 360 degrees and in OpenCV this range is comprised from 0 to 180 degrees. Through the function findContour in OpenCV, it is possible to detect elementary geometrical forms as squares, rectangles, circle, etc. and it is also possible to define proper own geometrical forms. In Figure 3 we show the result we obtained by considering as objective the form-color detection of a can.

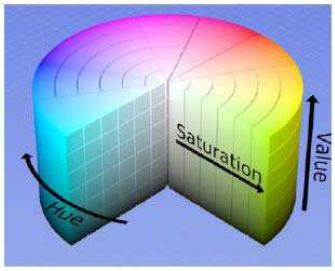



Fig. 1: HSV model.


Fig. 2: RGB vs HSV.

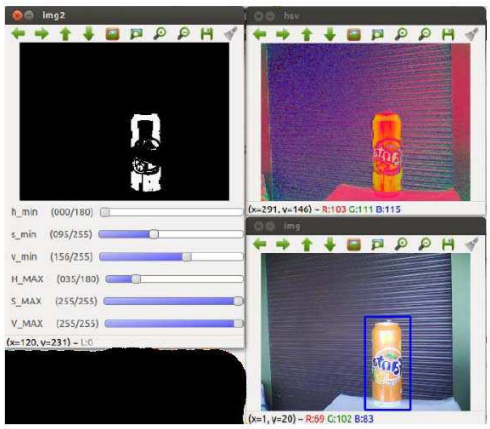



Fig. 3: Identification Process of a Can.

**Face Recognition based on cascade Classificators** Face Recognition is an example of high-level recognition and is one of the most complicated examples of pattern individuation. The mathematical representation of a face is really complicated when compared to elementary geometric as triangles, circles, etc.

Fig. 4: Identification of an Human Face. @Lena.

In order to be able to realize the goal of Face Recognition, we need to formulate it as a *Machine Learning* problem, where an initial training phase is mandatory. In OpenCV, it is also possible to find some configuration to detect specific parts of a face, stored in XML files. If something different is necessary, it is possible to train the Classifier by using a proper own set of images with the specific object that need to be detected. The training phase needs to be realized in a very accurate way, since many different factors need to be taken into account, such as the brightness, quality, color, etc. More specifically, the training phase we considered is called *HaarTraining*, [4]. This procedure requires a lot of computational resources, since the training phase requires the analysis of many images to extract the information necessary to create a classifier. The operation is based on some intermediate phases, based on the analysis of "negative" and "positive" images. Specifically, the "positive" images are those that contain inside the target that has to be identified. The "negative" images are those without the target to be identified. The term "cascade" is referred to the fact that the classifier is the result of various simple classifiers (phases), that are applied in a sequential way either to a frame or to an image. In practice, the output of a classifier is the input of the next classifier and so on. In Figure 4, we can observe the result of searching a human face inside an image, by applying the Cascade Classifiers method. It is thus very efficient.

## 4 Adaptation of Tools

This section presents how the tools described in Section 3 are modified to make the robots able to detect in an effective way a target object by considering information coming both from the surrounding (i.e. WebCam) and the other Rovers (by enabling communication among them). In this section, we will detail how the tools described above have been modified in order to be suitable with our distributed devices. It is worth noting that the libraries of OpenCV have

been conceived for powerful centralized systems whereas we are considering distributed devices. We focus on two specific techniques:

- Object Detection based on HSV model;
- Face Recognition based on Cascade Classifiers.

The first fundamental step, for both techniques, consists in the choice of the right filter to analyze images/frames. In OpenCV various filters are available such as Canny, Gaussian, Kalman, ConDensation, etc. In order to test the different filters on our Rovers, we modeled them as Python models and we implemented them on the devices.

### 4.1 HSV Object Detection

The Object Detection task consists in a first phase named Detection Phase. The frame coming from the WebCam passes through a filter and has to be opportunely resized, due to the scarce computational resources of our device. After the resizing, the frame will be converted from RGB to HSV. It is also necessary to determine the characteristic values of a color. This problem has been faced by realizing a suitable interface as shown in Figure 7.

Resizing an image makes the detection more difficult, but by iteratively applying a specific OpenCV filter that enlarges the image, we can overcome this issue. Unfortunately, the impurities of the color spectrum will be enlarged too. In order to solve this additional problem, we apply the *GaussianBlur* filter that makes the image more homogeneous. After these basic operations, the target definition algorithm can be integrated in the Behavioral Algorithm and is ready to be tested. The test phase shows an additional problem related to the "retrieving" of the right HSV values from the WebCam. This data generates a stream video that has to be sent to the GUI. By taking the limited resources of our device into account, we had to apply a MPEG encoding to the stream, whereas originally data was YUV. This encoding change alters the correct HSV values. It is also worth recalling that the HSV technique is based on the brightness and contrast concepts. In order to obtain a mechanism able to work in every environmental condition, we need powerful hardware, but we will show in Section 5, that in certain conditions (places with low light and with constant brightness), the algorithm implemented on our Rover works in a very effective way.

### 4.2 Cascade Classifiers Face Recognition

In order to realize the Face Recognition task, we formulate it as a *Machine Learning* problem and we consider a training phase named *HaarTraining* [9]. In the training phase, many factors need to be taken into account in a very accurate way, such as brightness, quality of the image, color, etc. The *HaarTraining* procedure requires a lot of computational resources, since the training phase is based on the analysis of many images in order to have the necessary information to realize a classifier. Concerning this training phase, we can distinguish the

analysis of "positive" images (that contains the target that has to be identified) and "negative" images (where the target is not detected). In order to perform the Face Recognition task, we consider a suitable OpenCV Classifier. Also in this case, as in the previous application of HSV, we need to manage the image by resizing it and we convert the RGB model to the GRAY model. This latter step is necessary in order to decrease the amount of information to process and to improve the speed of the analysis. The next step consists in the integration of the data into the Behavioral Algorithm. After this integration, we performed some tests and we noticed that the movements of the Rovers and data of the Task seemed misaligned. Once again, the scarce processing resources have made a correct processing of data frames, impossible. In order to fix this problem, we reduced the number of frames processed in the unit of time. Instead of considering a continuous data flow, we limited the number of frames to 15, for each operation. In practice, we analyzed 15 frames at the center, 15 frames at left and 15 frames at right. Moreover, we included a "release" phase for the WebCam after each acquiring, that implies the presence of "settling" frames, where the WebCam exploits its focus functions to define the image. In Section 5, we experimentally determine the number of frames useful for this purpose and that have to be discarded from the analysis of the image. Another kind of issue is related to the "false positive" detection, namely some object that is not a face is wrongly identified as a target. In order to detect false positives, we modified the Recognition Algorithm, by considering "trustable" a detection where at least 3 over 10 frames recognize an object as target.

In summary, we can claim that the tools we considered for managing multimedia data, opportunely modified in order to work in an effective way with the Rovers, are really effective for detection tasks.

### 4.3 Implementation

In this section we describe how the various phases of the Behavioral Algorithm have been implemented into our Rover by using our adapted tools. The main actions performed by our devices are detailed in Figure 5.

**Obstacle detection and bypassing** As mentioned in Section 2, an obstacle detection and bypassing is underlying and run by every moving robot. Obstacle detection is performed through the use of the robot ultrasound sensors. At every time, the robot checks whether it detects an object by applying the mechanism described in Section 4.1. If this object (identified as an obstacle if not the target) is close, the robot has to bypass it. To do so, the robot just turns $\alpha^\circ$ right. If the obstacle is still there, the robot turns $2 \times \alpha^\circ$ left. If the obstacle is still there, the robot moves backward, turns $\alpha^\circ$ right again and resumes its previous movement (either in searching or approaching phase).

**Searching Phase** After a preliminary check of the obstacle presence, the Rover is able to perform the Searching Phase, by acquiring the data frames in a suitable

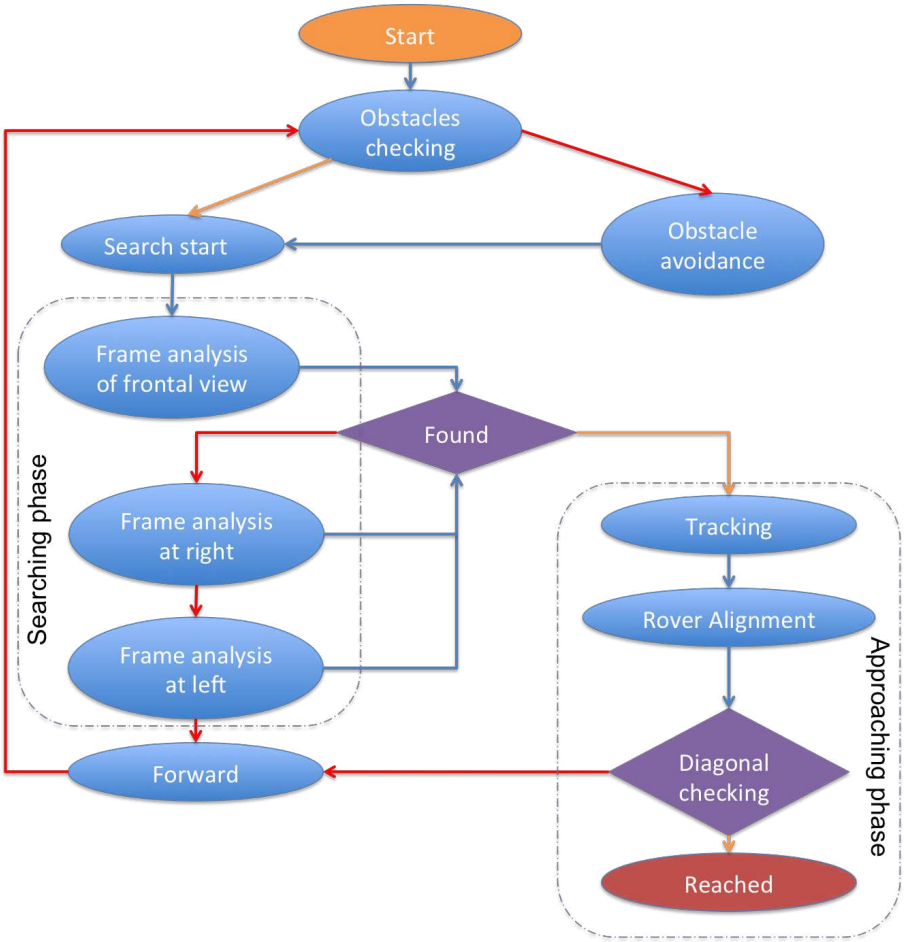



Fig. 5: Rover actions Flowchart.

way. Data acquisition is performed as Whether the robot detects the target, it will start to reach it. The searching phase is performed through the processing of the WebCam frames. From the analysis of the frames, the device makes a comparison with the target, and will be able to establish whether the target has been found or not. Objets are detected by using adapted tool described in Section 4.1 and identified by using the face recognition tool as explained in Section 4.2.

**Approaching Phase** In this phase, the robot simply heads in direction of the target till either reaching it or realizing that the target is a "false positive". To detect "false positive", at every step, the robot simply runs the face recognition tools detailed in Section 4.2 again to check whether it confirms the detected object in the right target. The target is considered as reached based on the size

Fig. 6: Target reachability. On the left side, the target is supposed unreached since the green diagonal length is under a threshold. On the right side, the target is considered to be close enough and reached.

of objects on the WebCam pictures. The robot stops when the diagonal length of the identified target is larger than a threshold as shown on Figure 6

Once the target is reached, the robot needs to advertise its peers to allow them to abort their current task and move to the next one in the list. To do so, the robot sends a radio beacon to its neighbors.

## 5 Testbed Description and Results

This section describes the performance of the Behavioral Algorithm implemented on our Rovers in order to accomplish Detection and Covering Tasks.

We consider two scenarios: the first one is based on the use of the HSV technique and the last is a Face Recognition Task in which the Rovers are also able to communicate to each other in order to exchange information about the tasks.

### 5.1 HSV Scenario

In this scenario the task, robots are assigned the detection of a specific object, namely the Rover has to move to towards the target. In Figure 7 one can observe the generation of the HSV Values, by using the tools of the HSV Gui.

When the HSV Task starts, the Behavioral Algorithm allows the process of the Searching Phase and Approaching Phase and the Rover tries to accomplish the assigned task. In this work, we were able to test the effectiveness of the algorithm implemented, by considering an environment with low light and constant brightness conditions. In Figure 8 we can observe a snapshot of the experiments conducted. In order to test the effectiveness of the algorithm, we considered many different scenarios, by keeping similar brightness conditions, and the Rover was able to accomplish its assigned task in all the tests.

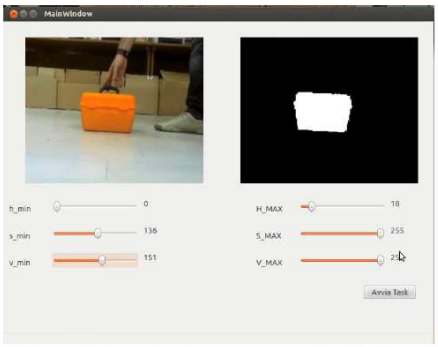



Fig. 7: HSV Gui Interface. The orange object is the target assigned.

Fig. 8: The target is reached.

### 5.2 Face Recognition Scenario

In this scenario the target to be detected is a human face.

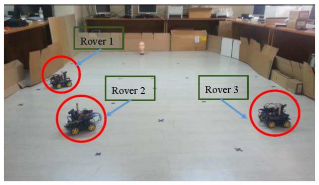
The goal of the test is the localization of the face and the first Rover that identifies and reaches the Target, sends an "alert" message to the others, in order to "attract" them. The other devices are required to dynamically change their task in this case. In Figure 9(a), we show the Laboratory room where the test has been realized.

We realized some tests to verify the exact number of frames used by the WebCam as "settling" frames, and we verified that 5 frames are necessary and have to be discarded. In practice, the first 5 frames cannot be analyzed.

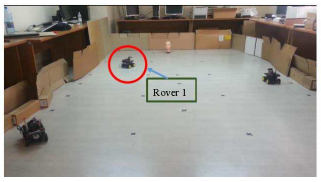
In the first phase, the Rovers start the Target Searching Phase (see Figure 9(b)).

In the specific test we are considering, the Rover 1 is the first one that detects the target, and then, it starts the Approaching Phase as shown in the Figure 10(a), whereas Rovers 2 and 3 continue in their searching phase. When Rover 1 reaches the target, it broadcasts a message to Rovers 2 and 3 by switching on its red led, in order to be visually identified as a target (see Figure 10).

(a) The room where the tests have been realized.

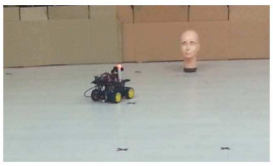



(b) Searching Phase for the three Rovers.

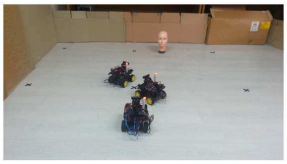



(c) Identification of the Target.

Fig. 9: The Phases to searc



(a) The led is switched on.



(b) The others two Rovers co-operate with the first one and reach it.

Fig. 10: Cooperation among the Rovers.

When Rovers 2 and 3 receive the message, they are able to dynamically move to the next task and, their objective will be modified (see Figure 10(b)).

We repeated this kind of tests many times, by changing the initial position of the Rovers and by modifying the position of the target. We noticed, that the only problem related with this kind of experiments was an excessive luminosity. In fact, in this case the limited number of frames elaborated by the devices is not enough accurate.

## 6 Conclusions and Future Works

In this work we faced the real implementation of mobile devices based on Arduino platforms and equipped with many sensors and a WebCam. The devices are able to accomplish Detection Tasks of objects and faces. The target objects can be identified by exploiting the combination of shape/color. On the Rovers we implemented a Behavioral Algorithm, where the commands related with the detection have been integrated. In order to make effective the tools considered for the detection purpose, we opportunely modified them, considering the scarce

resources available and the movement of the Rovers. We realized a proof-of-concept of our Behavioral Algorithm and we have shown its effectiveness in different scenarios. The main issue is related with the brightness conditions of the environment and with rapid changing of the luminosity, but the use of more powerful computational devices allows to overcome easily this kind of matter. As future work, we will consider quad-core mini-pc, and we will test our Behavioral Algorithm in more variable conditions of luminosity.

## References


1. http://www.arduino.cc/, accessed on-line on 23-August-2013.
2. G. Beni, "From swarm intelligence to swarm robotics," in Swarm Robotics Workshop: State-of-the-Art Survey, E. Sahin and W. Spears, Eds., no. 3342, pp. 1-9, Springer, Berlin, Germany, 2005.
3. L. Blzovics, K. Csorba, B. Forstner, C. Hassan, "Target Tracking and Surrounding with Swarm Robots," Engineering of Computer-Based Systems, IEEE International Conference on the, pp. 135-141, 2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems, 2012
4. http://note.sonots.com/SciSoftware/haartraining.html
5. http://en.wikipedia.org/wiki/$HSL_and_HSV$
6. E. Hyytiä, J. Virtamo, "Random waypoint model in n-dimensional space", in Operations Research Letters, 2005.
7. K.K. Jinasena and R.G.N. Meegama, "Design of a Low-cost Autonomous Mobile Robot", in International Journal of Robotics and Automation, (IJRA), vol. 2, issue 1, 2011.
8. H.-B. Kuntze, C. W. Frey, I. Tchouchenkov, B. Staehle, E. Rome, K. Pfeiffer, A. Wenzel, J. Wollenstein, "SENEKA - Sensor Network with Mobile Robots for Disaster Management," in Proceedings of IEEE Conference on Technologies for Homeland Security (HST), 13–15 November 2012.
9. R. Lienhart, J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," in IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002. http://www.lienhart.de/ICIP2002.pdf
10. R. Lundh, L. Karlsson, A. Saffiotti,"Autonomous Functional Configuration of a Network Robot System," in Robotics and Autonomous Systems, vol. 56, pp. 819–830, 2008
11. J. Nagi, G. A. Di Caro, A. Giusti, L. Gambardella, Convolutional Support Vector Machines for quantifyingl the visual learning and recognition progress in swarm robotic systems, Proceedings of the 11th International Conference on Machine Learning and Applications (ICMLA 2012), Boca Raton, Florida, December 12-15, 2012
12. opencv.org
13. C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, T. Stirling, A. Gutierrez, L.M. Gambardella and M. Dorigo, "ARGoS: a Modular, Multi-Engine Simulator for Heterogeneous Swarm Robotics", in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, USA, September 25-30, 2011
14. A. Sanfeliu, N. Hagita, A. Saffiotti, "Robotics and Autonomous Systems", in Elsevier Robotics and Autonomous Systems, vol. 56, pp. 793–797, July 2008.